5Ф

УДК 519.667

Л.П. Фельдман, Д.А. Завалкин
Донецкий национальный технический университет, г. Донецк, Украина
feldman@r5.dgtu.donetsk.ua, dimzav@gmail.com

Эффективность и масштабируемость параллельных одношаговых блочных методов решения задачи Коши

В статье рассматриваются два семейства параллельных неявных блочных методов решения задачи Коши. Определяется порядок аппроксимации каждого из методов, дается оценка для невязок методов. Для предложенного метода распараллеливания производится построение оценок ускорения, эффективности и масштабируемости. На основании полученных оценок даются рекомендации, при каких условиях данные методы наиболее продуктивны.

Введение

Применение высокопроизводительных параллельных вычислительных систем (ВС) является одним из основных направлений развития современной компьютерной науки. Это обстоятельство вызвано не только принципиальным ограничением максимально возможного быстродействия обычных последовательных машин, но и постоянным появлением новых вычислительных задач, для решения которых возможностей существующих средств вычислительной техники всегда оказывается недостаточно. Моделирование реальных многомерных динамических процессов, описываемых системами обыкновенных дифференциальных уравнений (СОДУ), и представляет собой класс задач, при реализации которых использование параллельных суперкомпьютеров не только оправдано, но и необходимо. Об этом свидетельствует известный список «большой вызов», в котором такие задачи занимают одно из ведущих мест [1].

Целью данной работы является анализ двух семейств классических параллельных одношаговых блочных методов, в том числе сравнительный анализ их характеристик: сначала осуществляется оценка невязок, на основе которой для методов с одинаковым порядком аппроксимации осуществляется оценка ускорения, эффективности и масштабируемости.

Задача Коши

Задача Коши для системы обыкновенных дифференциальных уравнений (СОДУ) первого порядка с известными начальными условиями имеет вид (1). Существует много параллельных методов решения задачи Коши для СОДУ первого порядка. В данной статье мы рассмотрим два метода – канонические одношаговые блочные методы и одношаговые блочные методы, предложенные Т.А. Биккартом в [2], в дальнейшем для краткости мы будем их называть методом «типа Биккарта».

$$\begin{cases} \dfrac{dy_1(x)}{dx} = f_1(x; y_1, y_2, ..., y_m), \quad y_1(x_0) = y_{10}, \\ \dfrac{dy_2(x)}{dx} = f_2(x; y_1, y_2, ..., y_m), \quad y_2(x_0) = y_{20}, \\ \quad ... \\ \dfrac{dy_m(x)}{dx} = f_m(x; y_1, y_2, ..., y_m), \quad y_m(x_0) = y_{m0}. \end{cases} \tag{1}$$

Прежде чем приступить к описанию методов, введем некоторые обозначения. Множество S точек равномерной сетки $\{t_q\}$, $q = \overline{1,Q}$ и $t_s = T$ с шагом τ разобьем на N блоков, содержащих k точек каждый, при этом $k \times N \le Q$. В каждом блоке введем номер точки $i = \overline{0,k}$ и обозначим через $t_{n,i}$ точку n блока с номером i. Точку $t_{n,0}$ назовем началом блока n, а $t_{n,k}$ – концом блока. Очевидно, что имеет место $t_{n,k} = t_{n+1,0}$. Условимся начальную точку $t_0 = t_{1,0}$ в блок не включать. При численном решении задачи Коши для каждого следующего блока новые k значений функции вычисляются одновременно.

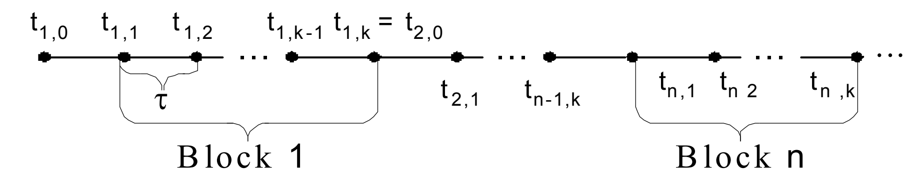


Рисунок 1 – Схема разбиения на блоки для одношагового k-точечного метода

Пусть $u_{n,0}$ – приближенное значение решения задачи Коши (1) в точке $t_{n,0}$ – начальной точке обрабатываемого блока. Предполагается, что в пределах одного блока точки сетки находятся на равных расстояниях: $t_{n,0} = t_n + i\tau, i = \overline{1,k}$.

Канонические одношаговые блочные методы

Уравнения канонических одношаговых разностных блочных методов для блока из k точек при использовании вычисленного значения приближенного решения в последнем предшествующем блоку узле, с учетом введенных выше обозначений можно записать в виде (2).

$$u_{n,i} = u_{n,0} + i \cdot ф \left[b_i \cdot F_{n,0} + \sum_{j=1}^{k} a_{i,j} \cdot F_{n,j} \right], i = \overline{1,k}, n = \overline{1,N}, \tag{2}$$

где $$F_{n,j} = f\left(t_n + j \cdot \tau, u_{n,j}\right).$$

Одношаговые блочные методы «типа Биккарта»

Уравнения одношаговых разностных блочных методов «типа Биккарта» для блока из k точек при использовании вычисленного значения приближенного решения в последнем предшествующем блоку узле, с учетом введенных выше обозначений можно записать в виде (3).

$$\frac{1}{\phi}\cdot\left[\sum_{j=0}^{k} a_{i,j}\cdot u_{n,j}\right]=F_{n,i},\ i=\overline{1,k},\ n=\overline{1,N}, \qquad (3)$$

где
$$F_{n,i}=f\left(t_n+i\cdot\tau,u_{n,i}\right).$$

Анализ невязок методов

Построим невязки для двухточечного канонического блочного метода (4) и двухточечного метода «типа Биккарта» (5).

$$\begin{aligned} r_{n,1}&=\frac{1}{24}x^{(4)}\left[t_n\right]\tau^3+O[\tau]^4 \\ r_{n,2}&=O[\tau]^4. \end{aligned} \qquad (4)$$

$$\begin{aligned} r_{n,1}&=\frac{1}{6}x^{(3)}\left[t_n\right]\tau^2+O[\tau]^3 \\ r_{n,2}&=-\frac{1}{3}x^{(3)}\left[t_n\right]\tau^2+O[\tau]^3 \end{aligned}. \qquad (5)$$

Из (4) и (5) видно, что двухточечный канонический блочный метод и двухточечный метод «типа Биккарта» имеют различные порядки аппроксимации, т.е. заведомо известно, что точность результатов этих двух методов будет различной. Для того чтобы сравнивать методы, нам необходимо, чтобы они имели один порядок аппроксимации. Возьмем трехточечный метод «типа Биккарта» (6).

$$\begin{aligned} r_{n,1}&=-\frac{1}{12}x^{(4)}\left[t_n\right]\tau^3+O[\tau]^4 \\ r_{n,2}&=\frac{1}{12}x^{(4)}\left[t_n\right]\tau^3+O[\tau]^4 \\ r_{n,3}&=-\frac{1}{4}x^{(4)}\left[t_n\right]\tau^3+O[\tau]^4 \end{aligned}. \qquad (6)$$

Как мы видим из (4) и (6), трехточечный метод «типа Биккарта» имеет тот же порядок аппроксимации, что и двухточечный канонический блочный метод. Аналогично можно проверить, что трехточечный канонический блочный метод и четырехточечный метод «типа Биккарта» имеют один порядок аппроксимации и т.д. Мы получаем, что у канонического блочного метода в блоке k точек, а у метода «типа Биккарта» $k+1$ точка. Введем условие, что у обоих методов размер блока одинаковый (рис. 2), тогда мы получим соотношение (7).

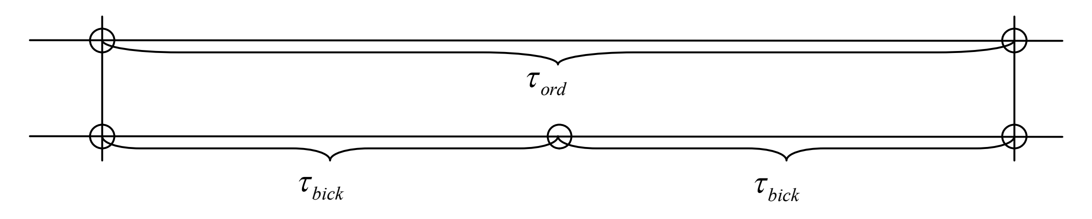


Рисунок 2 – Вычислительные сетки для двухточечного канонического блочного метода и трехточечного метода «типа Биккарта»

$$\left.\begin{aligned} k_{ord}\cdot\tau_{ord} &= k_{bick}\cdot\tau_{bick} \\ k_{bick} &= k_{ord}+1 \end{aligned}\right\} \Rightarrow \tau_{bick} = \frac{k_{ord}}{k_{ord}+1}\tau_{ord} = \frac{k}{k+1}\tau_{ord}\,. \qquad (7)$$

Общий вид уравнения невязки имеет вид (8).

$$r = C\cdot x^{k+1}(t_n)\cdot\tau^k\,,$$

где

$$C = \max_{i=\overline{1,k}}\left|C_i\right|. \qquad (8)$$

Из (8) для методов одного порядка аппроксимации с учетом (7) отношение невязок двух методов имеет вид (9).

$$\frac{r_{ord}}{r_{bick}} = \frac{C_{ord}}{C_{bick}}\cdot\frac{\tau_{ord}^k}{\tau_{bick}^k} = \frac{C_{ord}}{C_{bick}}\cdot\frac{\tau_{ord}^k}{\left(\frac{k}{k+1}\cdot\tau_{ord}\right)^k} = \frac{C_{ord}}{C_{bick}}\cdot\left(\frac{k}{k+1}\right)^k. \qquad (9)$$

Отношения невязок методов для $k = 2-6$ представлены в табл. 1.

Таблица 1 – Отношения невязок методов для $k = 2-6$

k	2	3	4	5	6
$\frac{r_{ord}}{r_{bick}}$	$\frac{3}{8}$	$\frac{4}{9}$	$\frac{1125}{4096}$	$\frac{99}{250}$	$\frac{4621925}{20155392}$

Так как мы положили, что блоки сетки в обоих методах равны, то в дальнейшем для упрощения записи мы можем дальнейшие вычисления и оценки делать не для n блоков сетки, а для одного блока.

Топологии сети передачи данных

Уравнения одношаговых разностных блочных методов «типа Биккарта» для блока из k точек при использовании вычисленного значения приближенного решения в последнем предшествующем блоку узле, с учетом введенных выше обозначений можно записать в виде (3).

Параллельные ВС имеют в своем составе поле процессоров, а также один или несколько модулей памяти. Эти функциональные компоненты должны быть связаны через соответствующие коммутационные системы. В силу специфики задач, решаемых на параллельных ВС [3], структура линий связи (топология сети передачи данных) должна удовлетворять различным требованиям. Во-первых, должна обеспечиваться связь между двумя любыми процессорами или модулями. Кроме этого, должно поддерживаться максимальное количество одновременных связей, чтобы средства коммутации не ограничивали параллельную обработку информации.

К числу типовых топологий обычно относят следующие схемы: линейка/кольцо, решетка/тор, гиперкуб [4]. Так как в последнее время наиболее распространены топологии решетка-тор и гиперкуб, мы будем проводить анализ методов для этих двух топологий.

Решетка (матрица, сетка – mesh) представляет собой систему, в которой процессоры расположены в виде правильной двумерной решетки и каждый процессор (кроме крайних) соединен с четырьмя соседями. Если в решетке граничные процессоры соединить линиями связи, то получится замкнутый вариант решетки или тор. Торроидальные 2D-схемы соединения имеют диаметр, пропорциональный $\sqrt{p}$ [5].

Структура гиперкуб (hypercube) является частным случаем решетки, когда по каждой размерности сетки имеется только два процессора [4]. То есть гиперкуб содержит $p = 2^N$ процессоров при размерности N. При гиперкубовой архитектуре число связей между процессорами небольшое: в N-мерном гиперкубе каждый процессор имеет N соседей.

Небольшим по сравнению с числом процессоров оказывается и диаметр системы, равный $\log_2 p$. Топология гиперкуб предоставляет возможность моделировать некоторые важные типы связей: линейка, кольцо, решетка. Недостатком описанной архитектуры является существование практического предела увеличения размерности гиперкуба.

Для оценки времени выполнения операции передачи одного сообщения объемом V байт между двумя процессами, локализованными на различных процессорах при распределенной памяти, используется следующая линейная модель, предложенная Хокни [6]:

$$T_{p-p} = t_s + t_w \cdot V \cdot l, t_w = \frac{y}{B}, \qquad (10)$$

где t_s – латентность, длительность подготовки сообщения для передачи;

l – длина маршрута;

t_w – время передачи одного байта;

y – число байт в слове;

B – пропускная способность канала передачи данных (байт/секунда).

Рассмотрим три коммуникационные операции:

1) передача данных между двумя процессорами сети – операция типа «точка-точка»;

2) передача данных от одного процессора всем остальным процессорам сети – операция «один-всем», или «one-to-all»;

3) передача данных от всех процессоров сети всем процессорам сети – множественная пересылка «все-всем», или «all-to-all».

Трудоемкость одиночной операции пересылки данных между двумя процессорами может быть получена путем подстановки длины максимального пути (диаметра сети) в выражение (10). Для вычисления времени выполнения множественной пересылки необходимо выбрать алгоритм маршрутизации. К числу наиболее распространенных оптимальных алгоритмов передачи данных относятся методы покоординатной маршрутизации [4]. Идея этих методов заключается в том, что поиск путей передачи данных осуществляется последовательно для каждой размерности рассматриваемой топологии.

Для топологии решетка-тор (рис. 3 а) одиночная операция пересылки данных требует следующего времени (11) для исполнения с учетом модели (10) и диаметра топологии.

$$T^M_{p-p} = t_s + 2V \cdot t_w \cdot \left\lfloor \sqrt{p} / 2 \right\rfloor. \qquad (11)$$

Пересылка данных от одного процессора всем остальным в условиях топологии тор может быть получена из этого же способа передачи для кольцевой топологии, выполненного в два этапа (12).

$$T^{M}_{one-to-all} = t_s + 2V \cdot t_w \cdot \left\lceil \sqrt{p} / 2 \right\rceil . \quad (12)$$

Множественная рассылка сообщений также может быть выполнена при помощи алгоритма, получаемого обобщением способа передачи данных для кольцевой структуры сети на основе идеи покоординатной маршрутизации (13).

$$T^{M}_{all-to-all} = 2t_s(\sqrt{p} - 1) + V \cdot t_w \cdot (p - 1) . \quad (13)$$

Для топологии гиперкуб (рис. 3 б) время выполнения одиночной операции пересылки данных равно (14).

$$T^{H}_{p-p} = t_s + V \cdot t_w \cdot \log_2 p . \quad (14)$$

Время выполнения операции передачи данных от одного процессора всем остальным в топологии гиперкуб выражается формулой (15).

$$T^{H}_{one-to-all} = (t_s + V \cdot t_w) \cdot \log_2 p . \quad (15)$$

Алгоритм выполнения множественной рассылки сообщений для гиперкуба может быть получен путем обобщения способа передачи данных «все-всем» для топологии решетка на размерность гиперкуба $lp = \log_2 p$. Каждому процессору ставится в соответствие двоичный эквивалент его номера. Процессоры, имеющие непосредственное соединение в гиперкубе, будут иметь номера, отличающиеся друг от друга только одним разрядом. На каждом этапе $i, i = \overline{1, lp}$ алгоритма функционируют все процессоры сети, которые обмениваются данными со своими соседями по i размерности и формируют объединенные сообщения (16).

$$T^{H}_{all-to-all} = \sum_{i=1}^{lp} (t_s + 2^{i-1} \cdot V \cdot t_w) = t_s \cdot \log_2 p + t_w \cdot V \cdot (p - 1) . \quad (16)$$

Алгоритм распараллеливания

Рассмотрим простейший в реализации метод распараллеливания – распараллеливание по уравнениям вычислительной схемы метода. Пусть в нашей вычислительной системе p процессоров, а система уравнений, для которой решается задача Коши, имеет m уравнений. Тогда на каждом процессоре в нашем алгоритме распараллеливания будут вычисляться $\frac{m}{p}$ уравнений для всех k точек текущего блока. После того как эти значения вычислены, каждый процессор рассылает значения в последней точке блока каждого своего уравнения остальным процессорам, т.е. происходит обмен типа «все-всем». Считая, что мы работаем с 8-байтными числами, согласно модели Хокни мы получаем время обмена для 1 блока сетки для топологии решетка-тор (17), а для топологии гиперкуб соответственно (18).

$$T^{M}_{all-to-all} = 2t_s(\sqrt{p} - 1) + 8 \cdot \frac{m}{p} \cdot t_w \cdot (p - 1) , \quad (17)$$

$$T^{H}_{all-to-all} = t_s \cdot \log_2 p + t_w \cdot 8 \cdot \frac{m}{p} \cdot (p - 1) . \quad (18)$$

Анализ ускорения и эффективности

Обозначим через m количество уравнений в исходной системе, через δ – время выполнения одной операции с плавающей точкой, а через $T_f(\delta)$ – сложность правой части, т.е. время вычисления $F(x,t)$. Будем считать, что для нахождения решения во всех k точках блока с заданной точностью достаточно k итераций. С учетом введенных обозначений и предположения, мы получим, что время последовательного вычисления всех точек одного блока каноническим блочным методом выражается формулой (19), а методом «типа Биккарта» – выражением (20).

$$T_s^{ord} = (\mathrm{T}_f(\delta)\cdot k + (2\cdot k+4)\cdot\delta)\cdot k\cdot k\cdot m, \tag{19}$$

$$T_s^{bick} = \left(\mathrm{T}_f(\delta) + \left(k^2 - 2\cdot k + 4\right)\cdot\delta\right)\cdot k\cdot k\cdot m. \tag{20}$$

Время параллельного вычисления всех точек одного блока с использованием предложенного алгоритма распараллеливания каноническим блочным методом выражается формулой (21) для топологии решетка-тор и (22) для топологии гиперкуб, а методом «типа Биккарта» – выражением (23) для топологии решетка-тор и (24) для топологии гиперкуб соответственно.

$$T_{p-mesh}^{ord} = (\mathrm{T}_f(\delta)\cdot k + (2\cdot k+4)\cdot\delta)\cdot k\cdot k\cdot\frac{m}{p} + 2\cdot \mathrm{t}_s\left(\sqrt{p}-1\right) + 8\cdot\frac{m}{p}\cdot \mathrm{t}_w\cdot(p-1), \tag{21}$$

$$T_{p-hcube}^{ord} = (\mathrm{T}_f(\delta)\cdot k + (2\cdot k+4)\cdot\delta)\cdot k\cdot k\cdot\frac{m}{p} + \mathrm{t}_s\cdot\log_2 p + 8\cdot\frac{m}{p}\cdot \mathrm{t}_w\cdot(p-1), \tag{22}$$

$$T_{p-mesh}^{bick} = \left(\mathrm{T}_f(\delta) + \left(k^2 - 2\cdot k + 4\right)\cdot\delta\right)\cdot k\cdot k\cdot\frac{m}{p} + 2\cdot \mathrm{t}_s\left(\sqrt{p}-1\right) + 8\cdot\frac{m}{p}\cdot \mathrm{t}_w\cdot(p-1), \tag{23}$$

$$T_{p-hcube}^{bick} = \left(\mathrm{T}_f(\delta) + \left(k^2 - 2\cdot k + 4\right)\cdot\delta\right)\cdot k\cdot k\cdot\frac{m}{p} + \mathrm{t}_s\cdot\log_2 p + 8\cdot\frac{m}{p}\cdot \mathrm{t}_w\cdot(p-1). \tag{24}$$

В случае простой правой части $T_f(\delta)$ ускорение и эффективность обоих методов очень малы, поэтому рассмотрим случай достаточно сложной правой части $T_f(\delta) = 1000\cdot\delta$. Значения δ, t_s, t_w были взяты из [4], они являются реальными значениями, измеренными для кластера Нижегородского государственного университета. Количество процессоров p возьмем равным 512.

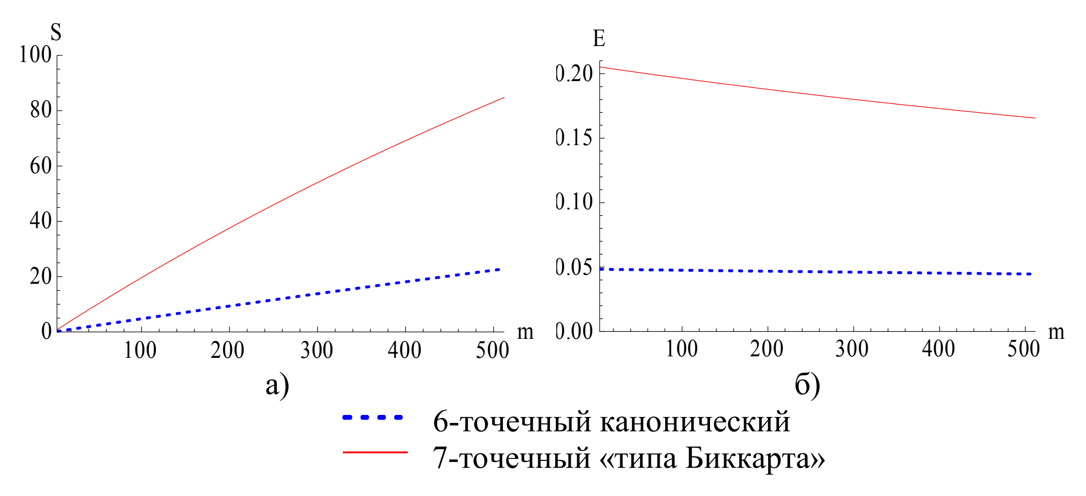


Рисунок 3 – Ускорение а) и эффективность б) канонического одношагового блочного метода и метода «типа Биккарта» для топологии решетка-тор

Построив графики ускорения и эффективности, легко увидеть, что для обеих рассматриваемых топологий и обоих рассматриваемых методов с ростом количества точек в методе улучшаются характеристики и ускорения, и эффективности. Сравним ускорение (рис. 3 а, рис. 4 а) и эффективность (рис. 3 б, рис. 4 б) наилучшего из представленных канонических одношаговых блочных методов и наилучшего из представленных методов «типа Биккарта» для обеих топологий.

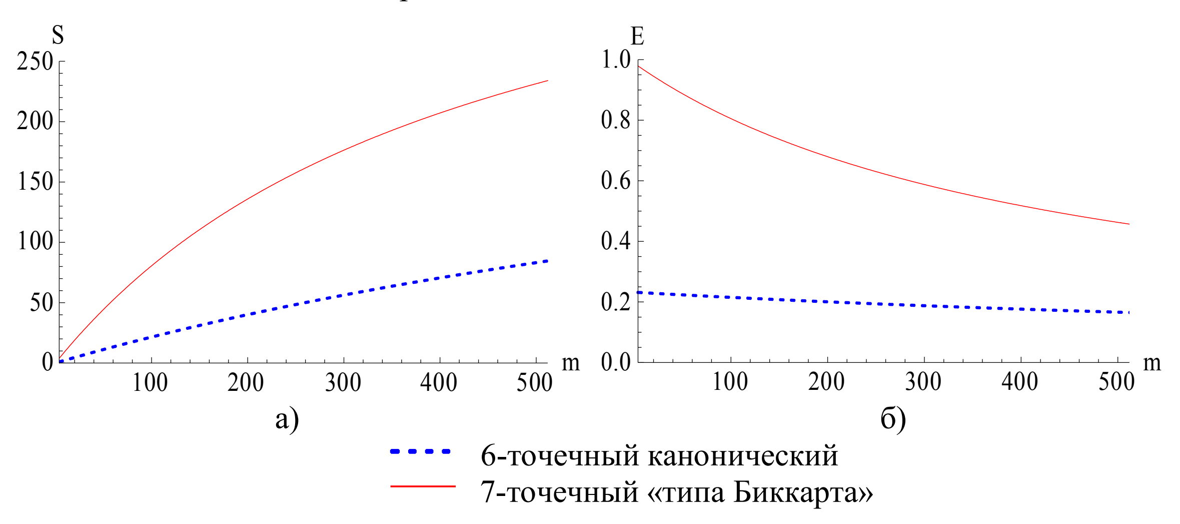


Рисунок 4 – Ускорение а) и эффективность б) канонического одношагового блочного метода и метода «типа Биккарта» для топологии гиперкуб

Анализ масштабируемости

Постановка вопроса об оценке эффективности распараллеливания при увеличении числа процессоров с сохранением объема вычислений не всегда является целью исследований, более того, некоторыми авторами [7] считается спорной. Действительно, при изменении числа процессоров эффективность параллельного алгоритма зависит как от свойств самой топологии сети, так и от того, насколько хорошо данный алгоритм подходит для данной топологии, насколько удачно он использует ее лучшие стороны. Поэтому нельзя исследовать масштабируемость параллельного алгоритма абстрактно, в отрыве от топологии сети.

Мы будем исследовать масштабируемость параллельных алгоритмов с помощью математического аппарата теории изоэффективного анализа [8], [9]. Для построения функции изоэффективности необходимо определить общие накладные расходы на параллельный алгоритм, для обоих алгоритмов они у нас будут одинаковыми вследствие выбранного алгоритма распараллеливания. Для топологии решетка-тор эти расходы выражаются формулой (25), а для топологии гиперкуб – формулой (26).

$$T_o^{mesh} = 2 \cdot \mathrm{t}_s \cdot p \cdot \left(\sqrt{p} - 1\right) + 8 \cdot m \cdot \mathrm{t}_w \cdot (p-1)\,, \qquad (25)$$

$$\mathrm{T}_o^{hcube} = \mathrm{t}_s \cdot p \cdot \log_2 p + 8 \cdot m \cdot \mathrm{t}_w \cdot (p-1)\,. \qquad (26)$$

Основное соотношение изоэффективного анализа [9] имеет вид (27).

$$T_s = K \cdot T_o\,, \qquad (27)$$

где

$$K = \frac{E}{1-E}\,. \qquad (28)$$

Тогда основное соотношение изоэффективного анализа для канонического одношагового блочного метода для топологии решетка-тор принимает вид (29), а для топологии гиперкуб – (30). Аналогично для метода «типа Биккарта» для топологии решетка-тор имеем (31), а для топологии гиперкуб – (32).

$$k^2 \cdot m \cdot (2 \cdot (2+k) \cdot \delta + k \cdot \mathrm{T}_f(\delta)) = K \cdot \left(2 \cdot \left(\sqrt{p} - 1\right) \cdot p \cdot \mathrm{t}_s + 8 \cdot m \cdot (p-1) \cdot \mathrm{t}_w\right), \quad (29)$$

$$k^2 \cdot m \cdot (2 \cdot (2+k) \cdot \delta + k \cdot \mathrm{T}_f(\delta)) = K \cdot \left(8 \cdot m \cdot (p-1) \cdot \mathrm{t}_w + p \cdot t_s \cdot \log_2 p\right), \quad (30)$$

$$k^2 \cdot m \cdot \left(\left(4 - 2 \cdot k + k^2\right) \cdot \delta + \mathrm{T}_f(\delta)\right) = K \cdot \left(2 \cdot \left(\sqrt{p} - 1\right) \cdot p \cdot \mathrm{t}_s + 8 \cdot m \cdot (p-1) \cdot \mathrm{t}_w\right), \quad (31)$$

$$k^2 \cdot m \cdot \left(\left(4 - 2 \cdot k + k^2\right) \cdot \delta + \mathrm{T}_f(\delta)\right) = K \cdot \left(8 \cdot m \cdot (p-1) \cdot \mathrm{t}_w + p \cdot t_s \cdot \log_2 p\right). \quad (32)$$

Так как количество точек в блоке $k = \overline{2,7}$ и мы рассматриваем случай систем с большим количеством уравнений $m >> k$ и сложной правой частью $T_f(\delta) >> k$, а также с учетом того, что мы полагаем $p < m$ всегда, то выражения (29) – (32) можно преобразовать к виду (33) – (36) соответственно.

$$\left[\begin{array}{l} m \cdot \mathrm{T}_f(\delta) \propto K \cdot m \cdot p \cdot t_w \\ m \cdot \mathrm{T}_f(\delta) \propto K \cdot p^{\frac{3}{2}} \cdot t_s \end{array}\right., \quad (33)$$

, (34)

$$\left[\begin{array}{l} m \cdot \mathrm{T}_f(\delta) \propto K \cdot m \cdot p \cdot t_w \\ m \cdot \mathrm{T}_f(\delta) \propto K \cdot p^{\frac{3}{2}} \cdot t_s \end{array}\right., \quad (35)$$

$$\left[\begin{array}{l} m \cdot \mathrm{T}_f(\delta) \propto K \cdot m \cdot p \cdot t_w \\ m \cdot \mathrm{T}_f(\delta) \propto K \cdot p \cdot \log_2 p \cdot t_s \end{array}\right.. \quad (36)$$

Как мы видим, для обоих видов топологий и обоих видов методов основное соотношение изоэффективного анализа принимает один и тот же вид (37).

$$m \cdot \mathrm{T}_f(\delta) \propto K \cdot m \cdot p \cdot t_w. \quad (37)$$

Из соотношения (37) следует, что при увеличении количества процессоров для сохранения постоянной эффективности можно увеличивать не только количество уравнений m системы, но и сложность правой части $\mathrm{T}_f(\delta)$.

Выводы

Канонические одношаговые блочные методы и методы «типа Биккарта» по причине использования одинакового метода распараллеливания обладают одинаковой масштабируемостью. Чем сложнее правая часть уравнения $\mathrm{T}_f(\delta)$, тем большее ускорение и эффективность показывают оба метода. Как видно из рис. 3 – 4 а) и б), ускорение и эффективность 6-точечного канонического блочного метода (наилучшего из рассмотренных канонических одношаговых методов) выше, чем ускорение и эффективность 7-точечного метода «типа Биккарта», при этом оба метода обладают одинаковым порядком аппроксимации. Как видно из табл. 1, при сравнении пар методов, обладающих одним по-

рядком аппроксимации, оказывается, что у канонического одношагового метода в каждой из пар невязка меньше, т.е. метод является более точным, в то же время его ускорение и эффективность выше, чем у метода «типа Биккарта» того же порядка аппроксимации. Учитывая это, а также то, что масштабируемость обоих методов одинакова за счет выбранного алгоритма распараллеливания, канонические одношаговые методы оказываются в целом лучше методов «типа Биккарта».

Литература

1. Grand Challenges: High performance computing and communications // A report by the Committee on Physical, Mathematical and Engineering Science, NSF/CISE, 1800 G. Street NW, Washington, DC 20550, 2001.
2. Sloate H.M. A-Stable Composite Multistep Methods / H.M. Sloate, T.A. Bickart // Journal of the Association for Computing Machinery. – 1973. – Vol. 20, № 1.
3. Гергель В.П. Основы параллельных вычислений для многопроцессорных вычислительных систем / В.П. Гергель, Р.Г. Стронгин. – Н. Новгород : ННГУ, 2001. – 122 с.
4. Гергель В.П. Теория и практика параллельных вычислений / Гергель В.П. – Москва : Бином. Лаборатория знаний, 2007. – 423 с.
5. Малиновский Б.Н. Вопросы построения высокопроизводительных средств обработки информации / Б.Н. Малиновский, В.П. Боюн, Л.Г. Козлов // Управляющие системы и машины. – 1976. – № 6. – С. 19-22.
6. Хокни Р. Параллельные ЭВМ: Архитектура, программирование и алгоритмы / Р. Хокни, К. Джессхоуп. – М. : Радио и связь, 1986. – 392 с.
7. Забродин А.В. Параллельные вычислительные технологии. Состояние и перспективы // Материалы I молодежной школы «Высокопроизводительные вычисления и их приложения» [Электронный ресурс] / А.В. Забродин. – Режим доступа : ftp://parallel.ru/parallel/chg1999.
8. Gupta A. Scalability of parallel algorithm for matrix multiplication / A. Gupta, V. Kumar // Technical report TR-91-54, Department of CSU of Minneapolis, 1994.
9. Kumar V. Scalability of Parallel Algorithms for the All-Pairs Shortest Path Problem / V. Kumar, V. Singh // Technical report, Department of CSU of Minneapolis, 1991.

Л.П. Фельдман, Д.О. Завалкін

Ефективність та здібність до масштабування паралельних блокових однокрокових методів вирішення задачі Коші

У статті розглядається дві сім'ї паралельних неявних блокових методів вирішення задачі Коші. Визначається порядок апроксимації кожного з методів, подається оцінка для відхилів методів. Для запропонованого методу розпаралелювання виконується побудова оцінок прискорення, ефективності та здібності до масштабування. На підставі отриманих оцінок даються рекомендації щодо умов, у яких подані методи найбільш продуктивні.

L.P. Feldman, D.A. Zavalkin

Efficiency and Scalability of Parallel Block One-Step Methods for Cauchy Problem Solution

Two families of parallel implicit block methods for Cauchy problem solution are considered in this paper. Approximation order and residual estimation of each method are found. For proposed parallelization algorithm parallel speedup, efficiency and scalability estimations are constructed. On the basis of derived estimations guidelines for conditions of method's using are proposed.

Статья поступила в редакцию 01.06.2009.